Feu tricolore intelligent à capteurs ultrasons Arduino Uno par ingénieur kambale kambere leonce

Code

```
// Feu Tricolore Intelligent 4 Voies - 2 Routes
// Gestion: File + Vitesse + Piétons
// Temps min: V=60s, J=20s, R=60s

// --- PINS ---
// Voie A Route 1: LEDs 22,23,24 | Capteurs: Trig 30 Echo 31 + Trig 32 Echo 33
// Voie B Route 1: LEDs 25,26,27 | Capteurs: Trig 34 Echo 35 + Trig 36 Echo 37
```

// Voie C Route 2: LEDs 28,29,30 | Capteurs: Trig 38 Echo 39 + Trig 40 Echo 41

// Voie D Route 2: LEDs 31,32,33 | Capteurs: Trig 42 Echo 43 + Trig 44 Echo 45

// Piétons: Bouton 2,3 | LED 46,47

// --- CONSTANTES TEMPS ---

const int T_VERT_MIN = 60000; // 60s

const int T_JAUNE_MIN = 20000; // 20s

const int T_ROUGE_MIN = 60000; // 60s

const int T_ADD_PAR_VOITURE = 3000; // +3s par voiture en plus

// --- CONSTANTES DETECTION ---

```
const int DIST_COMPTAGE = 200; // cm. Zone de comptage
const int DIST_VITESSE = 400; // cm. Distance entre 2 capteurs pour vitesse
const int VITESSE_CORTEGE_BASSE = 10; // km/h deuil/mariage
const int VITESSE_CORTEGE_HAUTE = 60; // km/h autorités

struct Voie {
  int ledV, ledJ, ledR;
  int trig1, echo1, trig2, echo2;
  int nbVoitures;
  bool priorite;
  unsigned long tempsVert;
};
```

```
Voie voies[4];
int voieActive = 0; // 0=A, 1=B, 2=C, 3=D
unsigned long chrono;

void setup() {
  Serial.begin(9600);
  initVoies();
  pinMode(2, INPUT_PULLUP); // Bouton pieton route 1
  pinMode(3, INPUT_PULLUP); // Bouton pieton route 2
  toutRouge();
  delay(2000);
  chrono = millis();
```

```
}

void loop() {
  // 1. MESURE
  for(int i=0; i<4; i++){
    voies[i].nbVoitures = compterVoitures(i);
    voies[i].priorite = detecterCortège(i);
    calculerTempsVert(i);
  }

  // 2. GESTION PIETONS
  if(digitalRead(2)==LOW ||
digitalRead(3)==LOW){
    gererPietons();
  }
```

```
  // 3. GESTION FEUX
  gererCycle();
}

// --- FONCTIONS PRINCIPALES ---

void gererCycle(){
  // Si priorité sur une voie, elle passe direct
  for(int i=0; i<4; i++){
    if(voies[i].priorite && i!=voieActive){
      passerAuVert(i);
      return;
    }
  }
```

```
  // Sinon cycle normal adaptatif
  if(millis() - chrono > voies[voieActive].tempsVert){
    int prochaineVoie = choisirProchaineVoie();
    passerAuVert(prochaineVoie);
  }
}

void passerAuVert(int nouvelleVoie){
  // 1. Orange sur voie actuelle
  setFeu(voieActive, 0,1,0);
  delay(T_JAUNE_MIN);
```

```
  // 2. Rouge sur voie actuelle
  setFeu(voieActive, 0,0,1);

  // 3. Attendre que l'autre route soit rouge depuis 60s
  int autreRoute = (nouvelleVoie<2)? 2 : 0; // Si A/B alors C/D et inverse
  delay(T_ROUGE_MIN);

  // 4. Vert sur nouvelle voie
  voieActive = nouvelleVoie;
  setFeu(voieActive, 1,0,0);
  chrono = millis();
}
```

```
void calculerTempsVert(int i){
  int base = T_VERT_MIN;
  int bonus = voies[i].nbVoitures *
T_ADD_PAR_VOITURE;
  voies[i].tempsVert = base + bonus;
  if(voies[i].tempsVert > 120000)
voies[i].tempsVert = 120000; // Max 2min
}

bool detecterCortège(int i){
  float vitesse = mesurerVitesse(i);
  if(vitesse > 0 && (vitesse <
VITESSE_CORTEGE_BASSE || vitesse >
VITESSE_CORTEGE_HAUTE)){
    Serial.print("PRIORITE VOIE ");
Serial.println(i);
```

```
    return true;
  }
  return false;
}

int choisirProchaineVoie(){
  // Choisir la voie avec le plus de voitures dans l'autre route
  int routeActuelle = (voieActive<2)? 0 : 2;
  int routeSuivante = (routeActuelle==0)? 2 : 0;

  int maxVoitures = -1;
  int meilleureVoie = routeSuivante;
```

```
  for(int i=routeSuivante; i<routeSuivante+2;
i++){
    if(voies[i].nbVoitures > maxVoitures){
      maxVoitures = voies[i].nbVoitures;
      meilleureVoie = i;
    }
  }
  return meilleureVoie;
}

// --- FONCTIONS CAPTEURS ---

int compterVoitures(int i){
  // Compte les fronts montants dans la
zone DIST_COMPTAGE
```

```
  // Simplifié ici. En vrai il faut anti-rebond
  return random(0,10); // A remplacer par vrai comptage
}

float mesurerVitesse(int i){
  // Mesure temps entre capteur1 et capteur2 séparés de DIST_VITESSE
  // vitesse = distance / temps
  return random(5,80); // A remplacer par vrai calcul
}

void gererPietons(){
  setFeu(voieActive, 0,1,0); // Orange
```

```
  delay(T_JAUNE_MIN);
  setFeu(voieActive, 0,0,1); // Rouge
  digitalWrite(46, HIGH); // LED pieton Vert
  delay(15000); // 15s pour traverser
  digitalWrite(46, LOW);
}

// --- UTILS ---

void setFeu(int voie, bool v, bool j, bool r){
  digitalWrite(voies[voie].ledV, v);
  digitalWrite(voies[voie].ledJ, j);
  digitalWrite(voies[voie].ledR, r);
}
```

```
void toutRouge(){
  for(int i=0; i<4; i++) setFeu(i,0,0,1);
}

void initVoies(){
  // Remplir la structure voies[] avec tous les pins
  //... à compléter avec tes pins exacts
}
```

Par ingénieur kambale kambere leonce